

# Compiler Project1

2017030519 홍유진

실행 환경은 Linux Ubuntu 이다.

<Tiny compiler modification>

1. 실행 방법

make 후

1\_Scanner 의 tinycompiler 디렉토리에 들어간후 컴파일

```
/1_Scanner/tinycompiler# ./tiny test.cm
```

2. 코드 설명

>>globals. H

```
#define MAXRESERVED 12

typedef enum
/* book-keeping tokens */
{ENDFILE,ERROR,
/* reserved words */
IF,ELSE, WHILE, RETURN, INT, VOID, THEN,END,REPEAT,UNTIL,READ,WRITE,
/* multicharacter tokens */
ID,NUM,
/* special symbols */

ASSIGN,EQ,LT,NE,LE,GT,GE,COMMA,LBRACE,RBRACE,LCURLY,RCURLY,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,SEMI
} TokenType;

extern FILE* source; /* source code text file */
```

새로운 symbol while, return, int, void, '==', '>=', '<=', '{', '}', '[', ']', '/\*'/'에 대한 enum 을 정의해줬다.  
그리고 MAXRESERVED 값도 12 로 변경해줬다.

>> main.c

```
/* set NO_PARSE to TRUE to get a scanner-only compiler */
#define NO_PARSE TRUE
/* allocate and set tracing flags */
int EchoSource = TRUE;
int TraceScan = TRUE;
int TraceParse = FALSE;
```

Scanner 를 제작할 수 있도록 변수값을 변경해주었다.

>> Util.c

```
case IF:
case THEN:
case ELSE:
case END:
case REPEAT:
case UNTIL:
case READ:
case WRITE:
case WHILE:
case RETURN:
case INT:
case VOID:
    fprintf(listing,
        "reserved word: %s\n",tokenString);
    break;
case ASSIGN: fprintf(listing,"=\n"); break;
case LT: fprintf(listing,"<\n"); break;
case EQ: fprintf(listing,"==\n"); break;
case NE: fprintf(listing,"!=\n"); break;
case LE: fprintf(listing,"<=\n"); break;
case GT: fprintf(listing,">\n"); break;
case GE: fprintf(listing,">=\n"); break;
case LBRACE: fprintf(listing,"{\n"); break;
case RBRACE: fprintf(listing,"}\n"); break;
case LCURLY: fprintf(listing,"[\n"); break;
case RCURLY: fprintf(listing,"]\n"); break;
case COMMA: fprintf(listing,",\n"); break;
case LPAREN: fprintf(listing,"(\n"); break;
case RPAREN: fprintf(listing,")\n"); break;
case SEMI: fprintf(listing,";\n"); break;
case PLUS: fprintf(listing,"+\n"); break;
case MINUS: fprintf(listing,"-\n"); break;
case TIMES: fprintf(listing,"*\n"); break;
case OVER: fprintf(listing,"/\n"); break;
case ENDFILE: fprintf(listing,"EOF\n"); break;
```

Reserved word 에 추가된 case 와 새로 추가된 symbol 들을 출력할 수 있도록 했다.

>>Scan.c

```
2f9f61b6:
{ 214b1'!'01'!INEQ' INCOMMENT' ININT'INID'DONE'ING'INCL'INIE'INOLEB'INCOMMENT'
f1b696: 0000
\* 2f9f62 fu 2c9006L DEL *\\
```

: state 를 나타내줄 enum 을 추가하였다. INEQ 는 ==을 INNE 는 !=을 INCOMMENT\_는 \*/을  
INCOMMENT 는 /\*를 INLT 와 INGT 는 >= <=를 알기 위해 추가되었다.'

```

/ Lookup table of reserved words /
static struct
{
    char* str;
    TokenType tok;
} reservedWords[MAXRESERVED]
= {{"if", IF}, {"else", ELSE}, {"while", WHILE}, {"return", RETURN}, {"int", INT}, {"void", VOID}, {"then", THEN}, {"end", END},
{"repeat", REPEAT}, {"until", UNTIL}, {"read", READ},
{"write", WRITE}};

```

: reservedword 를 추가해주었다.

```

switch (state)
{
    case START:
        if (isdigit(c))
            state = INNUM;
        else if (isalpha(c))
            state = INID;
        else if (c == '\n')
            state = INEQ;
        else if ((c == ' ') || (c == '\t') || (c == '\n'))
            save = FALSE;
        else if (c == '>')
            state = INGT;
        else if (c == '<')
            state = INLT;
        else if (c == '/')
            save = FALSE;
            state = INOVER;
        else if (c == '!')
            state = INNE;
    }
}

```

: 각 토큰들이 다른 토큰이 될 가능성이 있는 토큰들의 상태를 정의해주었다. INEQ 는 = 이거나 ==, INGT 는 > 이거나 >=, INLT 는 < 이거나 <=, /는 나누기/이거나 /\*\*/ !=이 될 가능성이 있다. 그래서 다음 토큰또한 보고 state 를 결정하기 때문에 state 를 우선 지정해주었다.

```

break;
case '=':
    currentToken = EQ;
    break;
case '<':
    currentToken = LT;
    break;
case '+':
    currentToken = PLUS;
    break;
case '-':
    currentToken = MINUS;
    break;
case '*':
    currentToken = TIMES;
    break;
case '(':
    currentToken = LPAREN;
    break;
case ')':
    currentToken = RPAREN;
    break;
case '[':
    currentToken = LCURLY;
    break;
case ']':
    currentToken = RCURLY;
    break;
case '{':
    currentToken = LBRACE;
    break;
case '}':
    currentToken = RBRACE;
    break;
case ';':
    currentToken = SEMI;
    break;
case ',':
    currentToken = COMMA;
    break;
default:
    break;

```

: 토큰 하나로 state 가 바로 결정될 수 있는 것들은 state 를 바로 지정해주었다.

```

case INNE:
    state = DONE;
    if (c == '=')
        currentToken = NE;
    else
        currentToken = ERROR;
    break;
case INCOMMENT:
    save = FALSE;
    if (c == EOF)
    {
        state = DONE;
        currentToken = ENDFILE;
    }
    else if (c == '/')
        state = START;
    else
        state = INCOMMENT;
    break;
case INGT:
    state = DONE;
    if (c == '=')
        currentToken = GE;
    else
    {
        ungetNextChar();
        currentToken = GT;
    }
    break;
case INEQ:
    state = DONE;
    if (c == '=')
        currentToken = EQ;
    else
    {
        currentToken = ASSIGN;
        ungetNextChar();
    }
    break;
case INLT:
    state = DONE;
    if (c == '<')
        currentToken = LE;
    else
    {
        ungetNextChar();
        currentToken = LT;
    }
    break;
case INOVER:
    if (c == '*')
    {
        state = INCOMMENT;
        save = FALSE;
    }
    else
    {
        ungetNextChar();
        state = DONE;
        currentToken = OVER;
    }
    break;

```

: 위의 정의한 state 들이 정확히 어떤 state 인지 판단할 수 있는 코드를 작성하였다.

/\*\*/는 처음 /를 보고 INOVER 로 인식 후 \*이 있으면 INCOMMENT 로 state 를 변경한다. State 가 INCOMMENT 이고 \*이 있으면 바깥쪽 주석처리하는 부분이므로 INCOMMENT\_로 state 를 변경한다. 그 후 state 가 INCOMMENT\_이면 /가 나오면 주석이 끝났다는 표현이므로 state 를 START 로 변경시킨다.

### 3. 실행 결과

- test.cm

```
gcc Math.0 util.0 scan.0 parse.0
root@hongyoujin:/home/hongyoujin/project1/:
TINY COMPILATION: test.cm
1: /* A program to perform []Euclid's
2:   Algorithm to computer gcd */
3:
4: int gcd(int u, int v)
4: reserved word: int
4: ID, name= gcd
4: (
4: reserved word: int
4: ID, name= u
4: ,
4: reserved word: int
4: ID, name= v
4: )
5: {
5: {
6: if(v<=0) return u;
6: reserved word: if
6: (
6: ID, name= v
6: <=
6: NUM, val= 0
6: )
6: reserved word: return
6: ID, name= u
6: ;
7: else return gcd(v,u-u/v*v);
7: reserved word: else
7: reserved word: return
7: ID, name= gcd
7: (
7: ID, name= v
7: ,
7: ID, name= u
7: -
7: ID, name= u
7: /
7: ID, name= v
7: *
7: ID, name= v
7: )
7: ;
8: /* u-u/v*v == u mod v */
9: }
10: void main(void)
11: reserved word: void
11: ID, name= main
11: (
11: reserved word: void
11: )
12: {
12: {
13: int x; int y;
13: reserved word: int
13: ID, name= x
13: ;
13: reserved word: int
13: ID, name= y
13: ;
14: x = input(); y = input();
14: ID, name= x
14: =
14: ID, name= input
14: (
14: ,
14: )
14: ;
14: ID, name= y
14: =
14: ID, name= input
14: (
14: )
14: ;
15: output(gcd(x,y));
15: ID, name= output
15: (
15: ID, name= gcd
15: (
15: ID, name= x
15: ,
15: ID, name= y
15: )
15: )
15: ;
16: }
16: }
17: EOF
```

-test2.cm

```
gcc Math.0 util.0 scan.0 parse.0
root@hongyoujin:/home/hongyoujin/p
TINY COMPILATION: test2.cm
1: void main(void)
1: reserved word: void
1: ID, name= main
1: (
1: reserved word: void
1: )
2: {
2: {
3: int i; int x[5];
3: reserved word: int
3: ID, name= i
3: ;
3: reserved word: int
3: ID, name= x
3: [
3: NUM, val= 5
3: ]
3: ;
4: i=0;
5: ID, name= i
5: =
5: NUM, val= 0
5: ;
6: while(i<5)
6: reserved word: while
6: (
6: ID, name= i
6: <
6: NUM, val= 5
6: )
7: {
7: {
8: ID, name= x
8: [
8: ID, name= i
8: =
8: ID, name= input
8: (
8: )
8: ;
9: i=i+1;
10: ID, name= i
10: =
10: ID, name= i
10: +
10: NUM, val= 1
10: ;
11: }
12: i=0;
13: ID, name= i
13: =
13: NUM, val= 0
13: ;
14: while(i<=4)
14: reserved word: while
14: (
14: ID, name= i
14: <=
14: NUM, val= 4
14: )
15: {
15: {
16: reserved word: if
16: (
16: ID, name= x
16: [
16: ID, name= i
16: ]
16: !=
16: NUM, val= 0
16: )
17: {
17: {
18: output(x[i]);
18: ID, name= output
18: (
18: ID, name= x
18: [
18: ID, name= i
18: ]
18: )
18: ;
19: }
19: }
20: }
21: }
22: EOF
```

<lex compiler modification>

## 1. 실행 방법

Makefile 을 수정한다.

```
OBJS = main.o util.o scan.o parse.o sytab.o analyze.o code.o cgen.o
OBJS_FLEX = main.o util.o lex.yy.o parse.o sytab.o analyze.o code.o cgen.o

.PHONY:all scanner_cimpl scanner_flex $(OBJS) $(OBJS_FLEX) lex.yy.c

all: scanner_cimpl scanner_flex

scanner_cimpl:$(OBJS)
$(CC) $(CFLAGS) $(OBJS) -o $@

scanner_flex:$(OBJS_FLEX)
$(CC) $(CFLAGS) $(OBJS_FLEX) -o $@ -lfl

lex.yy.o:cminus.l
flex $^
$(CC) $(CFLAGS) -c lex.yy.c
```

Make 후 1\_Scanner/lexcompiler 디렉토리로 들어간 후

```
./scanner_flex test.cm
```

## 2. 코드 설명

>> Globals.h util.h

: tiny compiler 와 동일하게 수정한다.

>> Cminus.l

```
"if"      {return IF;}
"then"    {return THEN;}
"else"    {return ELSE;}
"end"     {return END;}
"repeat"  {return REPEAT;}
"until"   {return UNTIL;}
"read"    {return READ;}
"write"   {return WRITE;}
"int"     {return INT;}
"while"   {return WHILE;}
"void"    {return VOID;}
"["       {return LBRACE;}
"]"       {return RBRACE;}
"return"  {return RETURN;}
"="       {return ASSIGN;}
"=="      {return EQ;}
"<"       {return LT;}
">"       {return GT;}
"<="      {return LE;}
">="      {return GE;}
"!="      {return NE;}
"+"       {return PLUS;}
"-"       {return MINUS;}
"*"       {return TIMES;}
"/"       {return OVER;}
"("       {return LPAREN;}
")"       {return RPAREN;}
";"       {return SEMI;}
{number}  {return NUM;}
{identifier} {return ID;}
{newline} {lineno++;}
{whitespace} { /* skip whitespace */ }
{"{"      {return LCURLY;}
"}"      {return RCURLY;}
","      {return COMMA;}

/*
 * Lexical analyzer for Cminus.
 */
{
    char c, tmp=NULL;
    do{
        c = input();
        if(c==EOF) break;
        if (c=='\n') lineno++;
        if(tmp=='*' && c=='/' ) break;
        tmp = c;
    }while(c);
    return ERROR;
}
```

새로운 symbol 들을 return 해준다. 이때 /\*/만 do-while 문을 이용하여 /\*/을 판단할 수 있도록 구현하였다.

### 3. 실행 결과

<test.cm>

```
root@hongyoujin:/home/hongyoujin/proje 11: reserved word: void
TINY COMPILATION: test.cm              11: ID, name= main
4: reserved word: int                  11: (
4: ID, name= gcd                        11: reserved word: void
4: (                                    11: )
4: reserved word: int                  12: {
4: ID, name= u                          13: reserved word: int
4: ,                                    13: ID, name= x
4: reserved word: int                  13: ;
4: ID, name= v                          13: reserved word: int
4: )                                    13: ID, name= y
5: {                                    13: ;
6: reserved word: if                    14: ID, name= x
6: (                                    14: =
6: ID, name= v                          14: ID, name= input
6: <=                                   14: (
6: NUM, val= 0                          14: )
6: )                                    14: ;
6: reserved word: return                14: ID, name= y
6: ID, name= u                          14: =
6: ;                                    14: ID, name= input
7: reserved word: else                  14: (
7: reserved word: return                14: )
7: ID, name= gcd                        14: ;
7: (                                    15: ID, name= output
7: ID, name= v                          15: (
7: ,                                    15: ID, name= gcd
7: ID, name= u                          15: (
7: -                                    15: ID, name= x
7: ID, name= u                          15: ,
7: /                                    15: ID, name= y
7: ID, name= v                          15: )
7: *                                    15: )
7: ID, name= v                          15: )
7: )                                    15: ;
7: ;                                    16: }
9: }                                    17: EOF
```

<test2.cm>

```
root@hongyoujin:/home/hongyoujin/proje 10: ID, name= i
TINY COMPILATION: test2.cm              10: =
1: reserved word: void                  10: ID, name= i
1: ID, name= main                       10: +
1: (                                    10: NUM, val= 1
1: reserved word: void                  10: ;
1: )                                    11: }
2: {                                    13: ID, name= i
3: reserved word: int                   13: =
3: ID, name= i                          13: NUM, val= 0
3: ;                                    13: ;
3: reserved word: int                   14: reserved word: while
3: ID, name= x                          14: (
3: [                                    14: ID, name= i
3: NUM, val= 5                          14: <=
3: ]                                    14: NUM, val= 4
3: ;                                    14: )
5: ID, name= i                          15: {
5: =                                    16: reserved word: if
5: NUM, val= 0                          16: (
5: ;                                    16: ID, name= x
6: reserved word: while                  16: [
6: (                                    16: ID, name= i
6: ID, name= i                          16: ]
6: <                                    16: !=
6: NUM, val= 5                          16: NUM, val= 0
6: )                                    16: )
7: {                                    17: {
8: ID, name= x                          18: ID, name= output
8: [                                    18: (
8: ID, name= i                          18: ID, name= x
8: ]                                    18: [
8: =                                    18: ID, name= i
8: ID, name= input                      18: ]
8: (                                    18: )
8: )                                    18: ;
8: ;                                    19: }
10: ID, name= i                         20: }
10: =                                   21: }
10: ID, name= i                         22: EOF
```